Esempio: Linear Congruential Generator

Si definisca una funzione:

```
def lcg(n, seed, a, c, m)
```

Che restituisca n elementi generati secondo la seguente ricorsione:

$$x_{n+1} = (ax_n + c) \mod m$$

Con $x_0 =$ `seed`

- Si utilizzino i valori $m = 16, a = 9, c = 3$
- Come *seed*, si utilizzi 0
- Si provino a generare sequenze di lunghezza crescente ($n = 4, 8, 12, \ldots, 24$)
  - Allo scopo, si utilizzi un ciclo

Esempio: Linear Congruential Generator

Di seguito una possibile soluzione

In [2]:
```
def lcg(n, seed, a, c, m):
    res = []
    x = seed # Elemento iniziale (da non restituire)
    for i in range(n):
        x = (a * x + c) % m # Ottengo il prossimo elemento
        res += [x] # Estendo la lista
    return res

for n in range(4, 25, 4):
    print(lcg(n, seed=0, a=9, c=3, m=16))
```

```
[3, 14, 1, 12]
[3, 14, 1, 12, 15, 10, 13, 8]
[3, 14, 1, 12, 15, 10, 13, 8, 11, 6, 9, 4]
[3, 14, 1, 12, 15, 10, 13, 8, 11, 6, 9, 4, 7, 2, 5, 0]
[3, 14, 1, 12, 15, 10, 13, 8, 11, 6, 9, 4, 7, 2, 5, 0, 3, 14, 1, 12]
[3, 14, 1, 12, 15, 10, 13, 8, 11, 6, 9, 4, 7, 2, 5, 0, 3, 14, 1, 12, 15, 10, 13, 8]
```

Esempio: Linear Congruential Generator

La ricorsione appena vista si chiama Linear Congruential Generator

...Ed ha alcune caratteristiche interessante:

- I numeri generati sono in un intervallo noto (da 0 a m)
- I numeri generati sono distribuiti uniformemente nell'intervallo
- Se non si conosce la regola, è difficile prevedere il numero successivo

Per questa ragione, possono esse trattati quasi come numeri casuali

Si parla per la precisione di numeri pseudo casuali

I.e. sembrano casuali, ma seguono una legge deterministica

- Intuitivamente, per un eleboratore elettronico è difficile comportarsi a caso
- ...Ma generara numeri casuali può essere molto utile (e.g. per fare simulazioni)!

I numeri pseudo-casuali permettono di aggirare in parte questa difficoltà

Esempio: Linear Congruential Generator

In pratical la nostra funzione è un Random Number Generator

Proviamo ad usarla per simulare un tiro di dado:

In [3]:
```
def dado(n, facce, seed=0):
    m, a, c = 2**16+1, 75, 74 # usiamo dei parametri un po' più seri
    res = lcg(seed=seed, n=n, m=m, a=a, c=c) # risultati "grezzi"
    return [v % facce for v in res]

facce = 6
risultati = dado(n=5, facce=facce, seed=42)
for v in risultati:
    print(f'Tiro un dado a {facce} facce! È uscito un {v}')
```

```
Tiro un dado a 6 facce! È uscito un 2
Tiro un dado a 6 facce! È uscito un 5
Tiro un dado a 6 facce! È uscito un 2
Tiro un dado a 6 facce! È uscito un 1
Tiro un dado a 6 facce! È uscito un 0
```